FaS 6차 세미나

애플리케이션 분석 실습 -Agit

20192233 박진철

CONTENTS

1. Kakao Agit 소개

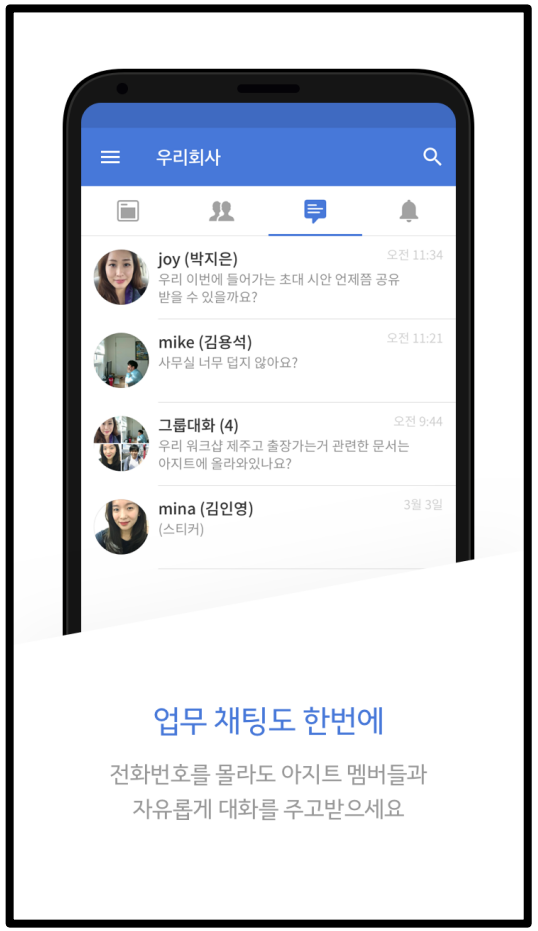
1. Kakao Agit 소개

- 카카오에서 2015년 개발
- 업무에서 활용가능한 기능이 들어있는 협업툴
- 채팅기능을 통해 팀원과 소통 가능

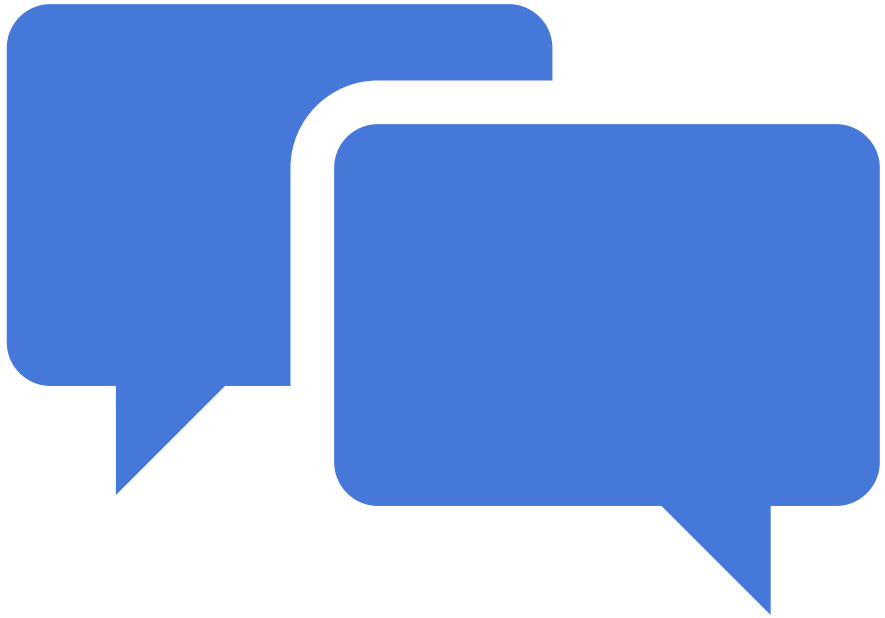
1. Kakao Agit 소개

- 카카오에서 2015년 개발
- 업무에서 활용가능한 기능이 들어있는 협업툴
- 채팅기능을 통해 팀원과 소통 가능

1. Kakao Agit 소개

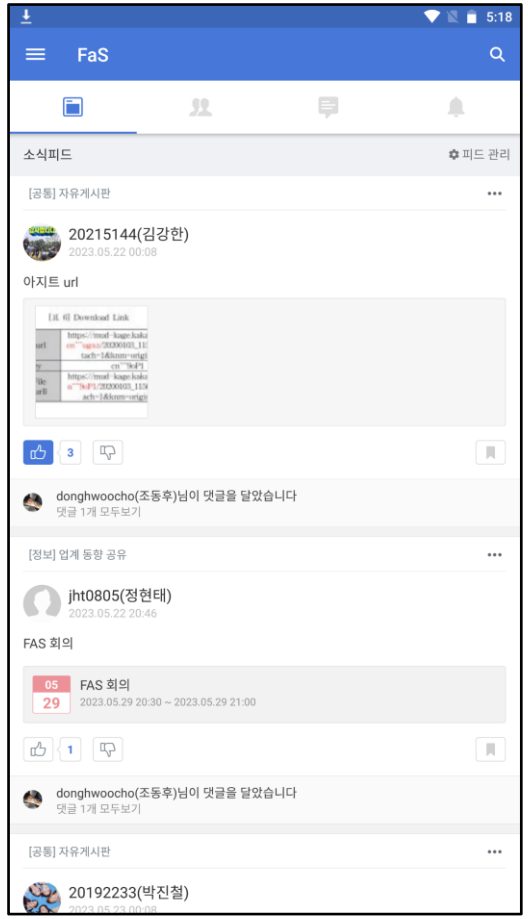


- 카카오에서 2015년 개발
- 업무에서 활용가능한 기능이 들어있는 협업툴
- 채팅기능을 통해 팀원과 소통 가능

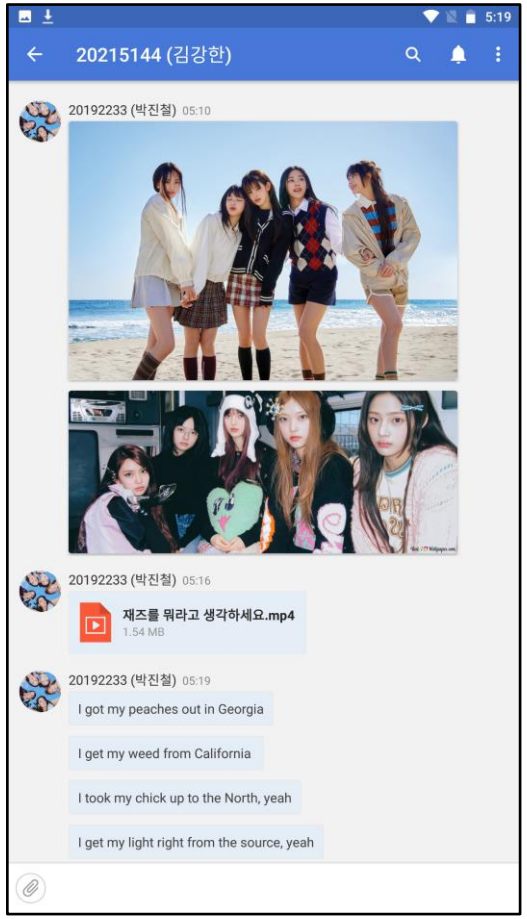


2. 아티팩트 적립과정

2. 아티팩트 적립과정

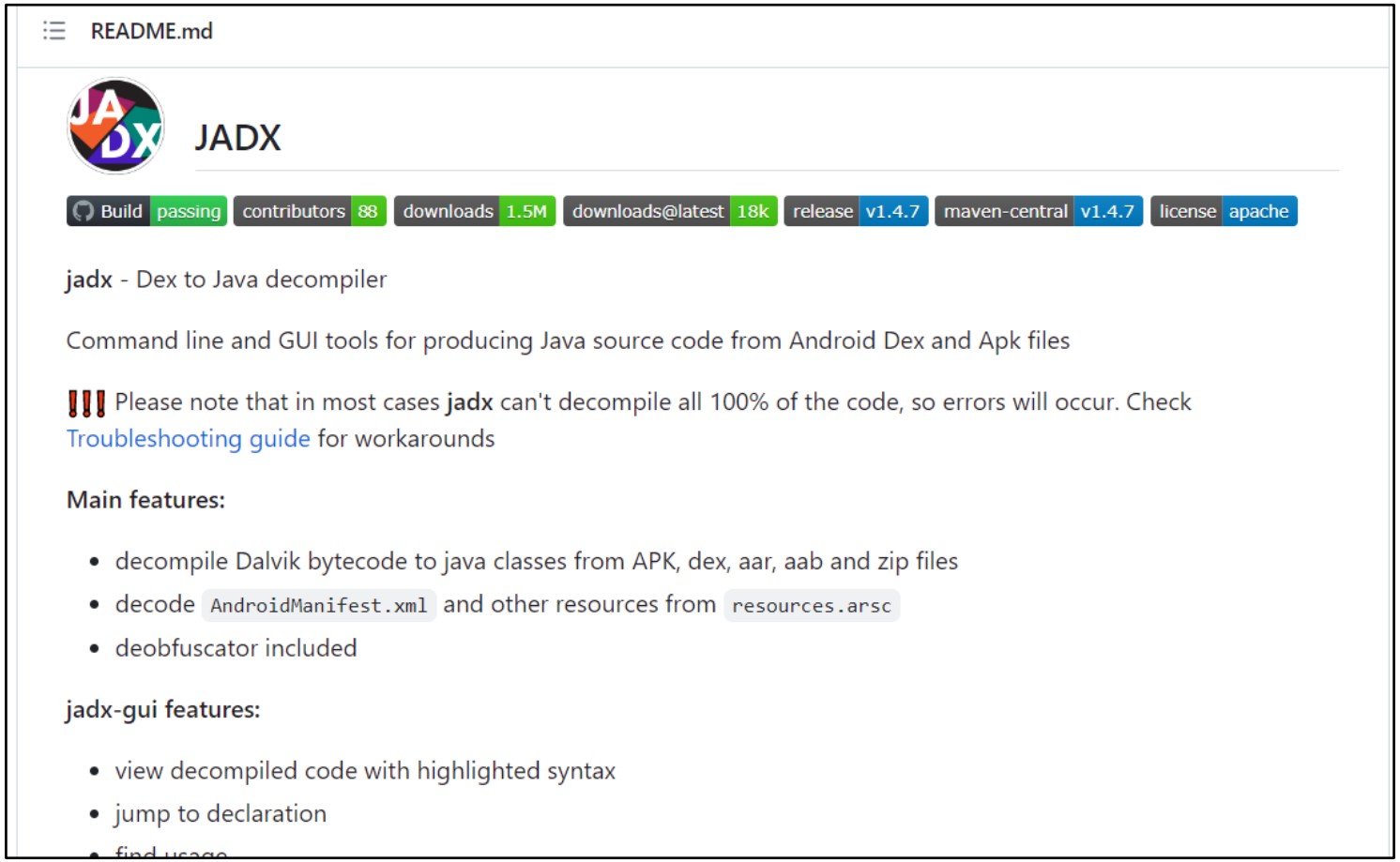


👉동아리에서 아지트 방 개설



👉채팅을 통해 다양한 메시지 전송

2. 아티팩트 적립과정



README.md

JADX

Build passing | contributors 88 | downloads 1.5M | downloads@latest 18k | release v1.4.7 | maven-central v1.4.7 | license apache

jadx - Dex to Java decompiler

Command line and GUI tools for producing Java source code from Android Dex and Apk files

❗❗❗ Please note that in most cases **jadx** can't decompile all 100% of the code, so errors will occur. Check Troubleshooting guide for workarounds

Main features:

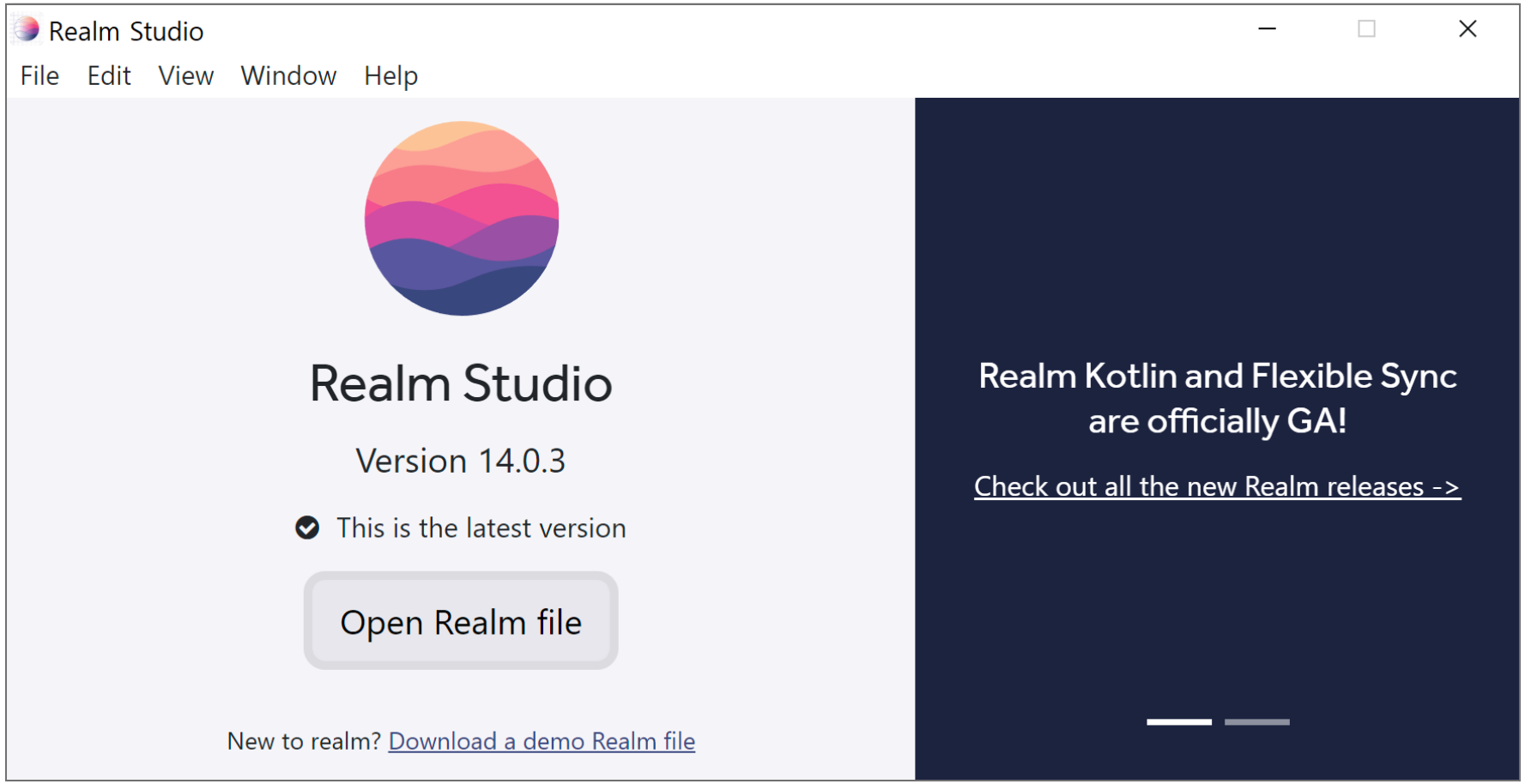
- decompile Dalvik bytecode to java classes from APK, dex, aar, aab and zip files
- decode `AndroidManifest.xml` and other resources from `resources.arsc`
- deobfuscator included

jadx-gui features:

- view decompiled code with highlighted syntax
- jump to declaration
- find usage

👉 jadx를 통해 APK파일 분석

2. 아티팩트 적립과정



👉 Realm Studio를 통해 realm 데이터 베이스 분석

3. 데이터 파일 분석

3. 데이터 파일 분석

- app_textures
- app_webview
- cache
- code_cache
- databases
- files
- lib
- no_backup
- shared_prefs

→

- application.preference.xml
- com.crashlytics.prefs.xml
- com.google.android.gms.appid.xml
- com.google.android.gms.measurement.prefs.xml
- group_list_cache.xml
- io.agit.xml
- io.agit_preferences.xml
- KakaoStory.hw.perferences.xml
- TwitterAdvertisingInfoPreferences.xml
- WebViewChromiumPrefs.xml

3. 데이터 파일 분석

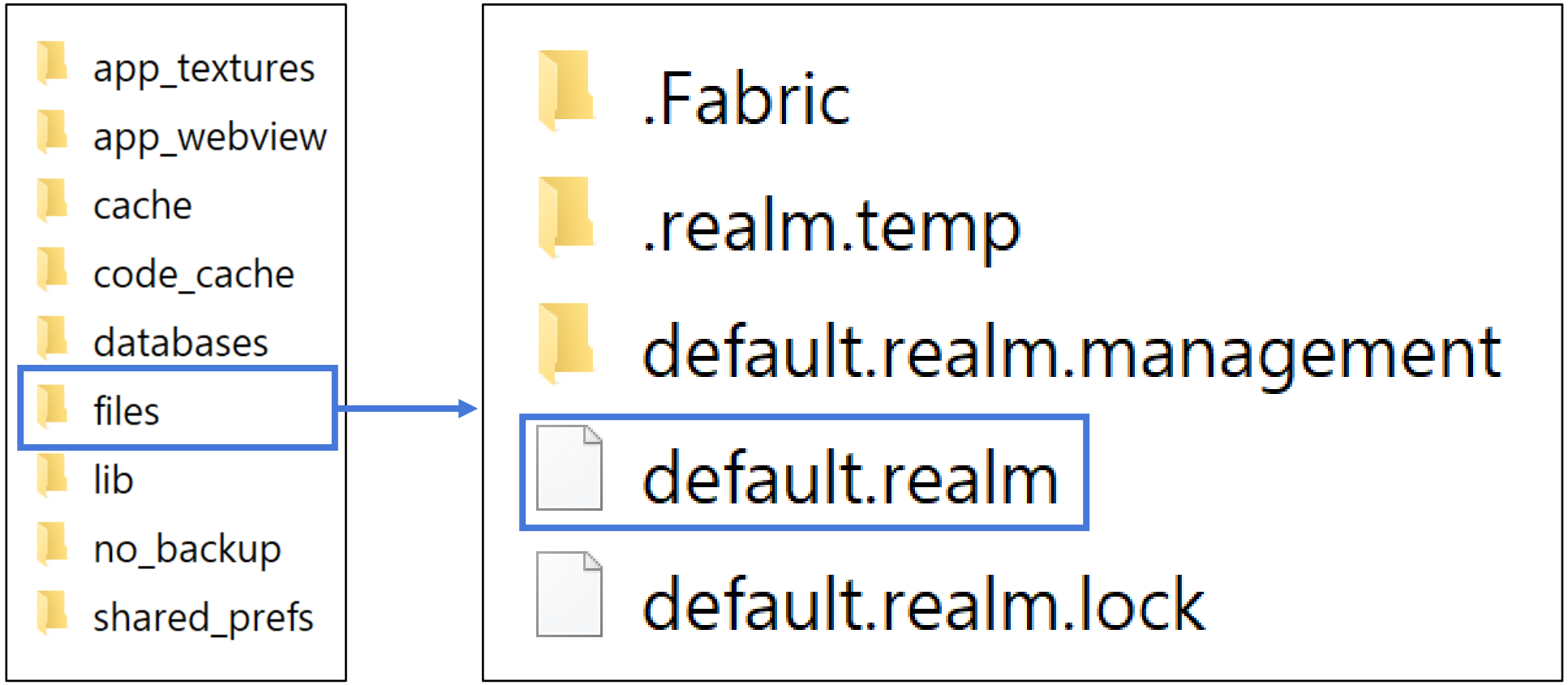
- application.preference.xml
- com.crashlytics.prefs.xml
- com.google.android.gms.appid.xml
- com.google.android.gms.measurement.prefs.xml
- group_list_cache.xml
- io.agit.xml
- io.agit_preferences.xml
- KakaoStory.hw.perferences.xml
- TwitterAdvertisingInfoPreferences.xml
- WebViewChromiumPrefs.xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
▼<map>
  <string name="user">
  {"email":"ic1595@kookmin.ac.kr","id":300241260,"birthDay":0,"birthMonth":0,"birthYear":0,"birthday":"
  암수 FaS","displayName":"20192233 (박진철)","mobilePhoneNumber":"01080003804","name":"박진
  철","permission":1,"planetHost":"fas.agit.io","deletedBot":false,"dummy":false,"guestUser":false,"hel
  암수 FaS","agit_id":"20192233","deactivated_at":false,"display_name":"20192233 (박진
  철)","is_bot":false,"is_current":false,"nickname":"박진철","on_vacation":false,"organizations":
  [],"permission_id":0,"planet":
  {"id":26839,"name":"FaS","masterId":-1,"subdomain":"fas","watermarkLevel":0,"downloadBlock":false,"en
  io/avatar/26839-300241260-ba9a4d02?dconv-s=SC&dconv-w=200&dconv-
  h=200","profile_image_url_large":"https://mk.kakaocdn.net/dn/agit-io/avatar/26839-300241260-
  ba9a4d02","status":{"expiration":0,"text":""},"user_detail":{"affiliation":"정암수
  FaS","birth_day":0,"birth_month":0,"birth_year":0,"mobile_phone":"01080003804","show_birthday":false,
  </string>
  <long name="user_list_modified_time" value="0"/>
  <boolean name="push_enabled" value="true"/>
  <string name="planet">
  {"id":26839,"name":"FaS","masterId":-1,"subdomain":"fas","watermarkLevel":0,"downloadBlock":false,"en
  </string>
  <long name="last_user_list_visited_time" value="0"/>
  <int name="sort_type" value="2"/>
</map>
```


👉 io.agit.xml을 통해 사용자 정보 확인 가능

3. 데이터 파일 분석



👉 files폴더에 default.realm 데이터베이스 존재

4. 데이터 베이스 복호화

4. 데이터 베이스 복호화

369

Journal of The Korea Institute of Information Security & Cryptology
VOL.30, NO.3, Jun. 2020

ISSN 1598-3986(Print)
ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2020.30.3.369

Realm 데이터베이스 암·복호화 프로세스 및 기반 애플리케이션 분석*

윤 병 철,[1†] 박 명 서,[1] 김 종 성[2‡]
[1,2]국민대학교(대학원생, 교수)

Analysis of Encryption and Decryption Processes of Realm Database and Its Application*

Byungchul Youn,[1†] Myungseo Park,[1] Jongsung Kim[2‡]
[1,2]Kookmin University (Graduate student, Professor)

요 약

모바일 기기의 보편화로 스마트폰 보급률 및 사용률이 계속해서 증가하고 있으며, 애플리케이션에서 저장 및 관리해야 할 데이터 또한 증가하고 있다. 최근 애플리케이션은 효율적인 데이터 관리를 위해 모바일 데이터베이스를 이용하는 추세이다. 2014년 개발된 Realm 데이터베이스는 지속적인 업데이트와 빠른 속도, 적은 메모리 사용, 코드의 간결함과 가독성 등의 장점을 바탕으로 개발자들의 관심이 증가하고 있다. 또한, 데이터베이스에 저장된 개인정보의 기밀성과 무결성을 제공하기 위해 암호화를 지원한다. 하지만 암호화 기능은 안티 포렌식 기법으로 사용될 가능성이 있으므로 Realm 데이터베이스가 제공하는 암·복호화 동작 과정 분석이 필요하다. 본 논문에서는 Realm 데이터베이스의 구조와 암·복호화 동작 과정을 상세히 분석하였으며, 분석 내용에 관한 활용 사례를 보이기 위해 암호화를 지원하는 애플리케이션을 분석하였다.

ABSTRACT

Due to the widespread use of mobile devices, smartphone penetration and usage rate continue to increase and there is

👉 해당 논문을 이용해 realm 데이터 베이스 복호화

4. 데이터 베이스 복호화

```
Input : Encrypted Realm Database, Encryption Key
Output : Decrypted Realm Database
Variable : Decrypted Realm Database Block = { D_0, D_1, ..., D_{n-1} }
           Encrypted Realm Database Block = { C_0, C_1, ..., C_{n-1} }
           IV_Table = iv1 || hmac1 || iv2 || hmac2

i ← 0
Pos ← 0
AES Key ← Upper 32 bytes of Encryption Key
HMAC Key ← Lower 32 bytes of Encryption Key
While( i < n) :
    Get IV_Table from Decrypted Realm Database
    if IV_Table.iv1 is 0 :
            Return False
    end if
    Compute hmac ← HMAC−SHA224( C_i, HMAC Key)
    if hmac and IV_Table.hmac1 is not equal
            if IV_Table.iv2 is 0 :
                    Return False
            end if
            if hmac and IV_Table.hmac2 is equal :
                    IV_Table.iv1 ← IV_Table.iv2
                    IV_Table.hmac1 ← IV_Table.hmac2
            else :
                    Return False
            end if
    end if
    Compute IV ← IV_Table.iv1 || Pos || 0 0 ... 0
    Decrypt D_i ← AES256−CBC( C_i, AES Key, IV)
    Write D_i in Decrypted Realm Database
    Pos ← Pos + 4096
    i ← i + 1
end While
Return Decrypted Realm Database
```

👉 논문에 들어있는 복호화 동작 과정

4. 데이터 베이스 복호화

```java
/* renamed from: a */
public C5867a m1961a(byte[] bArr) {
    if (bArr != null) {
        if (bArr.length == 64) {
            this.f23363d = Arrays.copyOf(bArr, bArr.length);
            return this;
        }
        throw new IllegalArgumentException(String.format(Locale.US, "The provided key must be %s bytes. Yours was: %s", 64, Integer.valueOf(bArr.length)));
    }
    throw new IllegalArgumentException("A non-null key must be provided");
}
```

```java
/* renamed from: i */
public final void m4078i() {
    RealmConfiguration.C5867a c5867a = new RealmConfiguration.C5867a();
    c5867a.m1961a(this.f21046h);
    c5867a.m1964a(11L);
    c5867a.m1962a(new DefaultMigration());
    this.f21047i = c5867a.m1965a();
}
```

```java
/* renamed from: h */
public byte[] f21046h = {80, 109, -111, -46, -111, -48, -44, -78, 78, -13, -5, Byte.MIN_VALUE,
```

4. 데이터 베이스 복호화

```java
/* renamed from: a */
public C5867a m1961a(byte[] bArr) {
    if (bArr != null) {
        if (bArr.length == 64) {
            this.f23363d = Arrays.copyOf(bArr, bArr.length);
            return this;
        }
        throw new IllegalArgumentException(String.format(Locale.US, "The provided key must be %s bytes. Yours was: %s", 64, Integer.valueOf(bArr.length)));
    }
    throw new IllegalArgumentException("A non-null key must be provided");
}
```

```java
/* renamed from: a */
public RealmConfiguration m1965a() {
    if (this.f23372m) {
```

⋮

```java
    String str = this.f23361b;
    return new RealmConfiguration(file, str, RealmConfiguration.m1987a(new File(file, str)), this.f23362c, this.f23363d, this.f23364e, this.f23365f, this.f23366g
}
```

```java
public RealmConfiguration(File file, String str, String str2, String str3, byte[] bArr, long j, RealmMigration realmMigration, boolean z, OsRealmConfig.EnumC6051b
    this.f23345d = file;
    this.f23346e = str;
    this.f23347f = str2;
    this.f23348g = str3;
    this.f23349h = bArr;
    this.f23350i = j;
    this.f23351j = realmMigration;
```

4. 데이터 베이스 복호화

```
public RealmConfiguration(File file, String str, String str2, String str3, byte[] bArr, long j, RealmMigration realmMigration, boolean z, OsRealmConfig.EnumC6051b
    this.f23345d = file;
    this.f23346e = str;
    this.f23347f = str2;
    this.f23348g = str3;
    this.f23349h = bArr;
    this.f23350i = j;
```

```
/* renamed from: e */
public byte[] m1981e() {
    byte[] bArr = this.f23349h;
    if (bArr == null) {
        return null;
    }
    return Arrays.copyOf(bArr, bArr.length);
}
```

```
public OsRealmConfig(RealmConfiguration realmConfiguration, boolean z, OsSchemaInfo osSchemaInfo, OsShar
    this.f24794e = new NativeContext();
```

⋮

```
    boolean equals2 = Boolean.TRUE.equals(m1863d[7]);
    byte[] m1981e = realmConfiguration.m1981e();
    if (m1981e != null) {
        nativeSetEncryptionKey(this.f24793d, m1981e);
    }
    nativeSetInMemory(this.f24793d, realmConfiguration.m1982d() == EnumC6051b.MEM_ONLY);
```

4. 데이터 베이스 복호화

Input : *Encrypted Realm Database*, *Encryption Key*
Output : *Decrypted Realm Database*
Variable : *Decrypted Realm Database Block* $= \{D_0, D_1, \ldots, D_{n-1}\}$
Encrypted Realm Database Block $= \{C_0, C_1, \ldots, C_{n-1}\}$
IV_Table $=$ *iv1* $\|$ *hmac1* $\|$ *iv2* $\|$ *hmac2*

```
i ← 0
Pos ← 0
AES Key ← Upper 32 bytes of Encryption Key
HMAC Key ← Lower 32 bytes of Encryption Key
While( i < n) :
    Get IV_Table from Decrypted Realm Database
    if IV_Table.iv1 is 0 :
            Return False
    end if
    Compute hmac ← HMAC−SHA224( C_i, HMAC Key)
    if hmac and IV_Table.hmac1 is not equal
            if IV_Table.iv2 is 0 :
                    Return False
            end if
            if hmac and IV_Table.hmac2 is equal :
                    IV_Table.iv1 ← IV_Table.iv2
                    IV_Table.hmac1 ← IV_Table.hmac2
            else :
                    Return False
            end if
    endif
    Compute IV ← IV_Table.iv1 ‖ Pos ‖ 00 ... 0
    Decrypt D_i ← AES256−CBC( C_i, AES Key, IV)
    Write D_i in Decrypted Realm Database
    Pos ← Pos + 4096
    i ← i + 1
end While
Return Decrypted Realm Database
```

```
/* renamed from: h */
public byte[] f21046h = {80, 109, -111, -46, -111, -48, -44, -78, 78, -13, -5,
```

👉 Encryption Key

4. 데이터 베이스 복호화

3.1.4 암호화된 Realm 데이터베이스 구조 분석

암호화된 Realm 데이터베이스의 구조는 IV_Table 컨테이너와 암호화된 블록 컨테이너로 나누어진다. 블록당 암호화 동작 과정이 종료되면 Realm 데이터베이스는 업데이트된 IV_Table과 암호화된 블록 순서로 쓰기 동작을 수행한다. 암호화된 Realm 데이터베이스의 구조는 Fig. 4와 같이 0x00 위치부터 4,096바이트 크기만큼 IV_Table을 차례로 저장하며, 이후 0x1000 위치부터 IV_Table 순서에 매칭되는 암호화된 블록을 저장한다.

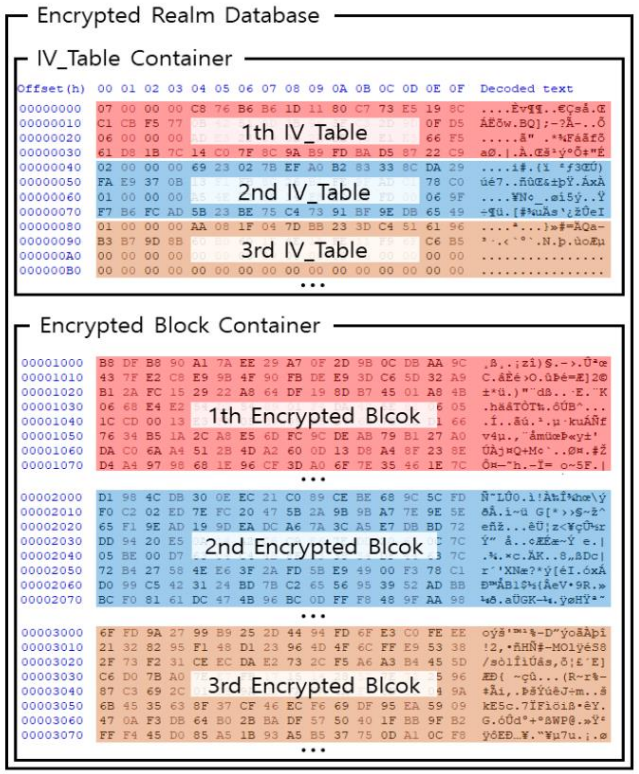


Fig. 4. Encrypted Realm Database Structure

Table 6. Structure of IV_Table

Offset	Size (Byte)	Field	Content
0x00	4	iv1	The first 4 bytes of IV using current encryption
0x04	28	hmac1	HMAC value of currently encrypted block
0x32	4	iv2	The first 4 bytes of IV using previous encryption
0x36	28	hmac2	HMAC value of previous encrypted block

4. 데이터 베이스 복호화

```python
from Crypto.Cipher import AES
#Input-Encryption Key
Enc_Key=[80, 109, 145, 210, 145, 208, 212, 178,
         78, 243, 251, 128, 225,  84,  40,  96,
         158, 180, 152, 122, 244, 144, 150, 223,
         154, 205, 126, 165, 173, 209,  70,  95,
         80, 141,  79,  71, 165, 158,  49,  52,
         207, 253,  14, 141, 215, 168,  11, 129,
         156, 117, 101, 193,  80, 229, 133, 129,
         72, 215, 245,  81, 233, 139,  58,  98]
#=>f21046h

#Input-Encrypted Realm Database
Enc_Realm_name="default.realm"
f = open(Enc_Realm_name, 'rb')
Enc_Realm = f.read()
f.close()
```

Input : *Encrypted Realm Database*, *Encryption Key*
Output : *Decrypted Realm Database*
Variable : *Decrypted Realm Database Block* $= \{D_0, D_1, \ldots, D_{n-1}\}$
Encrypted Realm Database Block $= \{C_0, C_1, \ldots, C_{n-1}\}$
IV_Table $= iv1 \| hmac1 \| iv2 \| hmac2$

```
i ← 0
Pos ← 0
AES Key ← Upper 32 bytes of Encryption Key
HMAC Key ← Lower 32 bytes of Encryption Key
While( i < n) :
    Get IV_Table from Decrypted Realm Database
    if IV_Table.iv1 is 0 :
        Return False
    end if
    Compute hmac ← HMAC−SHA224( C_i, HMAC Key)
    if hmac and IV_Table.hmac1 is not equal
        if IV_Table.iv2 is 0 :
            Return False
        end if
        if hmac and IV_Table.hmac2 is equal :
            IV_Table.iv1 ← IV_Table.iv2
            IV_Table.hmac1 ← IV_Table.hmac2
        else :
            Return False
        end if
    end if
    Compute IV ← IV_Table.iv1 || Pos || 00 ... 0
    Decrypt D_i ← AES256−CBC( C_i, AES Key, IV)
    Write D_i in Decrypted Realm Database
    Pos ← Pos + 4096
    i ← i + 1
end While
Return Decrypted Realm Database
```

4. 데이터 베이스 복호화

```
#Pos<-0
Pos=0

#Aes Key<-Upper 32 bytes of Encryption Key
AES_Key=bytes(Enc_Key[:32])

Dec = []
```

Input : *Encrypted Realm Database*, *Encryption Key*
Output : *Decrypted Realm Database*
Variable : *Decrypted Realm Database Block* $= \{D_0, D_1, \ldots, D_{n-1}\}$
Encrypted Realm Database Block $= \{C_0, C_1, \ldots, C_{n-1}\}$
IV_Table $=$ *iv1* $\|$ *hmac1* $\|$ *iv2* $\|$ *hmac2*

$i \leftarrow 0$
$Pos \leftarrow 0$
$AES\ Key \leftarrow Upper\ 32\ bytes\ of\ Encryption\ Key$
$HMAC\ Key \leftarrow Lower\ 32\ bytes\ of\ Encryption\ Key$
$While(i < n)$:
  Get IV_Table from Decrypted Realm Database
  if *IV_Table.iv1 is* 0 :
    Return False
  end if
  Compute hmac $\leftarrow HMAC-SHA224(C_i, HMAC\ Key)$
  if *hmac* and *IV_Table.hmac1 is not equal*
    if *IV_Table.iv2 is* 0 :
      Return False
    end if
    if *hmac* and *IV_Table.hmac2 is equal* :
      IV_Table.iv1 $\leftarrow$ *IV_Table.iv2*
      IV_Table.hmac1 $\leftarrow$ *IV_Table.hmac2*
    else :
      Return False
    end if
  end if
  Compute IV $\leftarrow$ *IV_Table.iv1* $\|$ *Pos* $\|$ 0 0 ... 0
  Decrypt $D_i \leftarrow AES256-CBC(C_i, AES\ Key, IV)$
  Write D_i *in Decrypted Realm Database*
  $Pos \leftarrow Pos + 4096$
  $i \leftarrow i + 1$
end While
Return Decrypted Realm Database

4. 데이터 베이스 복호화

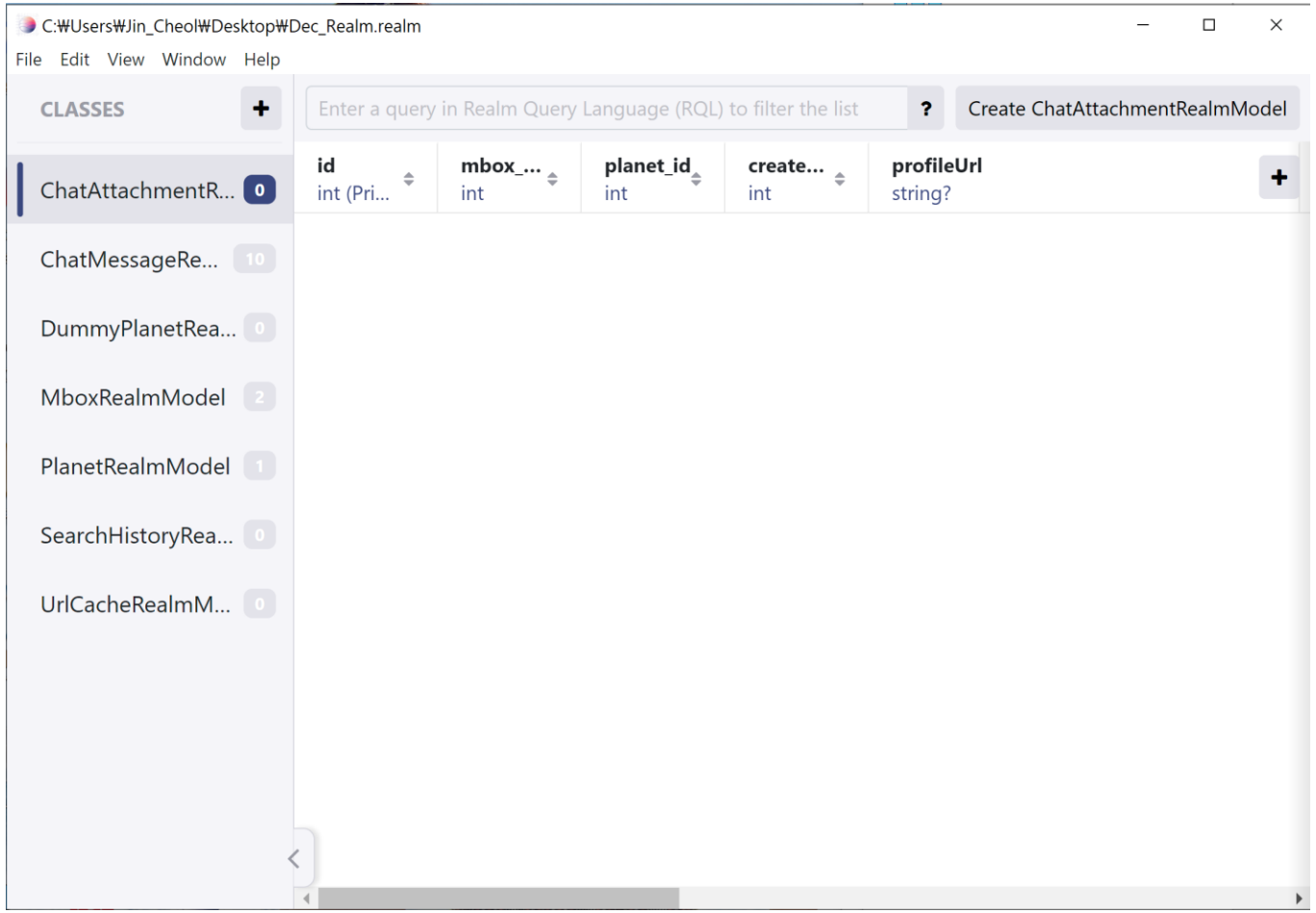
```python
#n: Database Block의 개수
n=len(Enc_Realm[0x1000:])// 4096
for i in range(0, n):
    #Get IV_Table from Decrypted Realm Database
    IV_Table=Enc_Realm[64*i:64*(i+1)]

    #Compute IV<-IV_Table.iv1||Pos||00000000
    IV=bytes(list(IV_Table[:4])+list((Pos).to_bytes(4,byteorder='big'))+[0]*8)

    #Decrypt D_i<-AES256-CBC(C_i, AES Key, IV)
    Enc_block=Enc_Realm[0x1000*(i+1):0x1000*(i+2)]
    cipher = AES.new(bytes(AES_Key), AES.MODE_CBC, IV)
    Dec_block=list(cipher.decrypt(Enc_block))

    #Write D_i in Decrypted Realm Database
    Dec += Dec_block

    #Pos<-Pos+4096
    Pos+=4096

f = open("Dec_Realm.realm", 'wb')
f.write(bytes(Dec))
f.close()
```

Input : *Encrypted Realm Database*, *Encryption Key*
Output : *Decrypted Realm Database*
Variable : *Decrypted Realm Database Block* $= \{D_0, D_1, \ldots, D_{n-1}\}$
Encrypted Realm Database Block $= \{C_0, C_1, \ldots, C_{n-1}\}$
IV_Table $= iv1 \| hmac1 \| iv2 \| hmac2$

```
i ← 0
Pos ← 0
AES Key ← Upper 32 bytes of Encryption Key
HMAC Key ← Lower 32 bytes of Encryption Key
While( i < n) :
    Get IV_Table from Decrypted Realm Database
    if IV_Table.iv1 is 0 :
        Return False
    end if
    Compute hmac ← HMAC−SHA224(C_i, HMAC Key)
    if hmac and IV_Table.hmac1 is not equal
        if IV_Table.iv2 is 0 :
            Return False
        end if
        if hmac and IV_Table.hmac2 is equal :
            IV_Table.iv1 ← IV_Table.iv2
            IV_Table.hmac1 ← IV_Table.hmac2
        else :
            Return False
        end if
    end if
    Compute IV ← IV_Table.iv1 || Pos || 00...0
    Decrypt D_i ← AES256−CBC(C_i, AES Key, IV)
    Write D_i in Decrypted Realm Database
    Pos ← Pos + 4096
    i ← i + 1
end While
Return Decrypted Realm Database
```

4. 데이터 베이스 복호화

Dec_Realm.realm	2023-05-29 오전 7:05	REALM 파일	144KB



5. 복호화된 메시지 확인

5. 복호화된 메시지 확인

C:₩Users₩Jin_Cheol₩Desktop₩Dec_Realm.realm

File Edit View Window Help

CLASSES

- ChatAttachmentR... 0
- ChatMessageRe... 10
- DummyPlanetRea... 0
- MboxRealmModel 2
- PlanetRealmModel 1
- SearchHistoryRea... 0
- UrlCacheRealmM... 0

Enter a query in Realm Query Language (RQL) to filter the list

id int (Pri...	mbox_... int	user_id int	message string?	created_time int
67822475	318736	300241587	안녕하세요!	1684755849
67823064	318736	30024126C	좋은 저녁입니다	1684767845
67823065	318742	30024126C	소가 칼을 물고 있으면?	1684768033
67823066	318742	30024126C	검문소	1684768035
67823067	318742	30024126C	깔깔깔깔	1684768039
67823106	318742	300241259	...? ㅋㅋㅋㅋㅋㅋㅋㅋㅋ	1684780122
67823107	318742	300241259	소가 불에 타면?	1684780158
67823108	318742	300241259	탄소	1684780160
67823109	318742	300241259	깔깔깔깔	1684780168
67861157	318742	30024126C	아이고 배꼽이야	1684858954

ter the list

profileUrl string?	displayName string?
https://t1.kakaocdn.net/agit_resources/ima	jht0805 (정현태)
https://mk.kakaocdn.net/dn/agit-io/avatar/	20192233 (박진철)
https://mk.kakaocdn.net/dn/agit-io/avatar/	20192233 (박진철)
https://mk.kakaocdn.net/dn/agit-io/avatar/	20192233 (박진철)
https://mk.kakaocdn.net/dn/agit-io/avatar/	20192233 (박진철)
https://t1.kakaocdn.net/agit_resources/ima	20215144 (김강한)
https://t1.kakaocdn.net/agit_resources/ima	20215144 (김강한)
https://t1.kakaocdn.net/agit_resources/ima	20215144 (김강한)
https://t1.kakaocdn.net/agit_resources/ima	20215144 (김강한)
https://mk.kakaocdn.net/dn/agit-io/avatar/	20192233 (박진철)

2023년 5월 23일 화요일

20192233 (박진철) 00:07

소가 칼을 물고 있으면?

검문소

깔깔깔깔

20215144 (김강한) 03:28

...? ㅋㅋㅋㅋㅋㅋㅋㅋㅋ

20215144 (김강한) 03:29

소가 불에 타면?

탄소

깔깔깔깔

2023년 5월 24일 수요일

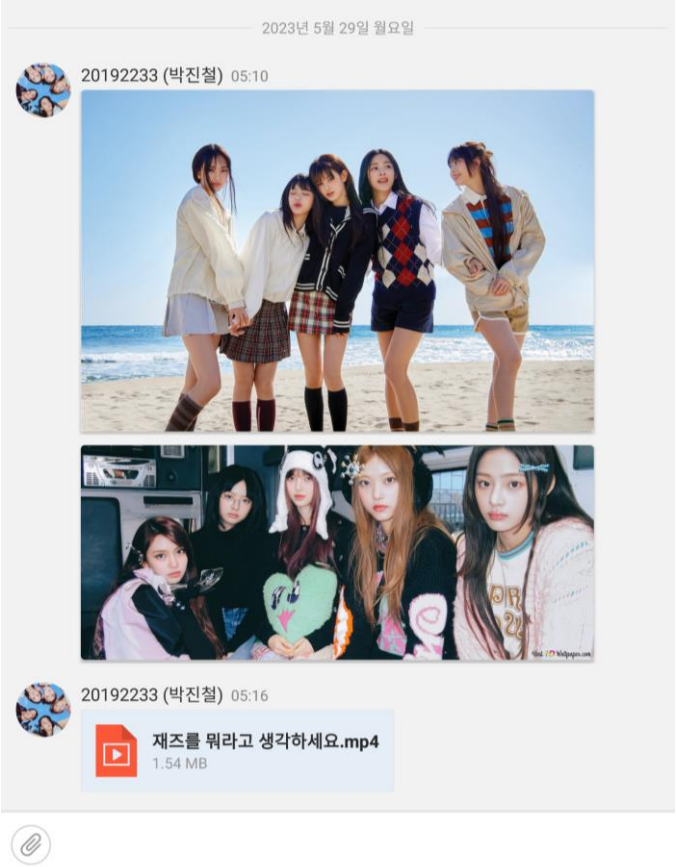
20192233 (박진철) 01:22

아이고 배꼽이야

20215144 (김강한) 01:42

ㅋㅋㅋㅋㅋㅋㅋㅋ

5. 복호화된 메시지 확인



👉 사진, 동영상 복호화를 하려 했으나
Realm파일이 열리지 않아서 성공하지 못했습니다..

THANKS TO WATCHING

20192233 박진철